An Improved Algorithm for the Periodic Timetabling Problem

Marc Goerigk[1] **and Christian Liebchen**[2]

1 **Department of Management Science, Lancaster University, Lancaster, UK**
m.goerigk@lancaster.ac.uk

2 **Technical University of Applied Sciences Wildau, Wildau, Germany**
liebchen@th-wildau.de

Abstract

We consider the computation of periodic timetables, which is a key task in the service design process of public transportation companies. We propose a new approach for solving the periodic timetable optimisation problem. It consists of a (partially) heuristic network aggregation to reduce the problem size and make it accessible to standard mixed-integer programming (MIP) solvers. We alternate the invocation of a MIP solver with the well-known problem specific modulo network simplex heuristic (ModSim). This iterative approach helps the ModSim-method to overcome local minima efficiently, and provides the MIP solver with better initial solutions.

Our computational experiments are based on the 16 railway instances of the PESPlib, which is the only currently available collection of periodic event scheduling problem instances. For each of these instances, we are able to reduce the objective values of previously best known solutions by at least 10.0%, and up to 22.8% with our iterative combined method.

1998 ACM Subject Classification G.1.6 Optimization, G.2.2 Graph Theory, G.2.3 Applications

Keywords and phrases periodic timetabling, railway optimisation, modulo network simplex, periodic event scheduling problem

Digital Object Identifier 10.4230/OASIcs.ATMOS.2017.12

1 Introduction

Railways play a central role in transportation. According to a report of the International Union of Railways, the largest operator in Germany, Deutsche Bahn, moved over 2 billion passengers and 79 billion passenger kilometers in 2015. However, between 2004 and 2014, inland passenger transport grew 5% slower than the constant price gross domestic product (GDP) in the EU-28[1].

One reason for this might be that not as many people as would be desirable from an ecological point of view are considering a journey by railway sufficiently attractive to make it their first choice. In particular in the absence of a direct trip, waiting times along transfers usually are highly disliked. Since transfer waiting times are an immediate outcome of the timetable, it is a major goal of railway companies to design a timetable that implies short transfer waiting times and hereby increases the attractiveness of their service offer.

In this paper we consider one such approach to increase the attractiveness of existing railway systems. To this end, we focus on railway networks that are operated periodically. Such a cyclic timetable repeats after the so-called period length T; e.g., it offers the same

[1] http://ec.europa.eu/eurostat/statistics-explained/index.php/Passenger_transport_statistics

17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017).
Editors: Gianlorenzo D'Angelo and Twan Dollevoet; Article No. 12; pp. 12:1–12:14
Open Access Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

services within each one-hour period. From a mathematical perspective, this property is reflected in the periodic event scheduling problem (PESP) as introduced in [14]. We are aiming at using PESP-based optimisation models and heuristics to construct periodic timetables with short waiting times, in particular at transfers.

Timetabling problems have been studied intensively over the last decades and much theoretical insight has been obtained. For a general survey on timetabling problems, we refer to [1] and to [2]. Optimised timetables have already been put into practice. In [9] it is described how mathematical optimisation was applied successfully to the Berlin Underground network: Keeping the very same number of trips offered, transfer times as well as dwell times of the trains at transfer stations had been reduced, while at the same time the timetable required one train unit less for operation. In the Netherlands, even the entire national railway network had been the subject of mathematical optimisation. Yet, even several further planning processes aside the actual timetable design had been included in this project, see [6]. The protagonists report an increase in both, passenger volume and punctuality, as well as several further improvements.

While the PESP model for cyclic timetabling is very flexible in capturing real-world constraints, see [10], it is also notoriously difficult to solve. The modulo network simplex (ModSim) heuristic [11, 4] is currently one of the strongest methods for large problem instances; recently, also a matching approach has been proposed [13] if the PESP does not contain limitations on the feasibility (see also [7]).

In this paper we present a new approach for solving the periodic timetabling problem, which includes the ModSim-method as a subroutine. It is based on the idea of a (partially) heuristic network aggregation to reduce the problem size. This allows the usage of a mixed-integer programming (MIP) solver in combination with ModSim. One advantage of such an approach is that the MIP solver and the heuristic have a completely different perspective on the problem structure, which allows to overcome local minima efficiently. Combinations of a MIP solver and a heuristic to overcome local optima have been successfully applied to other problems, such as the travelling tournament problem [5]. Our method considerably outperforms the ModSim-method as a stand-alone approach. Using instances of the PESPlib library[2], we are able to improve objective values of all current best solutions by at least 10.0%, and up to 22.8%.

The remainder of the paper is structured as follows. In Section 2 we formally introduce the periodic timetabling problem, before we present the network aggregation procedure in Section 3. In Section 4 we describe how we combine the ModSim-method with a standard MIP solver, based on the (heuristically) aggregated network, and report our experimental results.

2 The Periodic Timetabling Problem

We now briefly introduce the periodic event scheduling problem. We assume a so-called event-activity network (EAN) $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ to be given. Each node $i \in \mathcal{E}$ corresponds to an event occurring with periodicity $T \in \mathbb{N}$, while an arc $a = (i, j) \in \mathcal{A}$ models an activity between two events. In the case of the periodic timetabling problem, nodes correspond to arrival and departure events of trains at stations. Arcs model driving (from departure to arrival) or waiting (from arrival to departure) activities of trains. Additional activities are used to model, e.g., transfers of passengers, waiting times of trains or security (headway)

[2] `http://num.math.uni-goettingen.de/~m.goerigk/pesplib/`

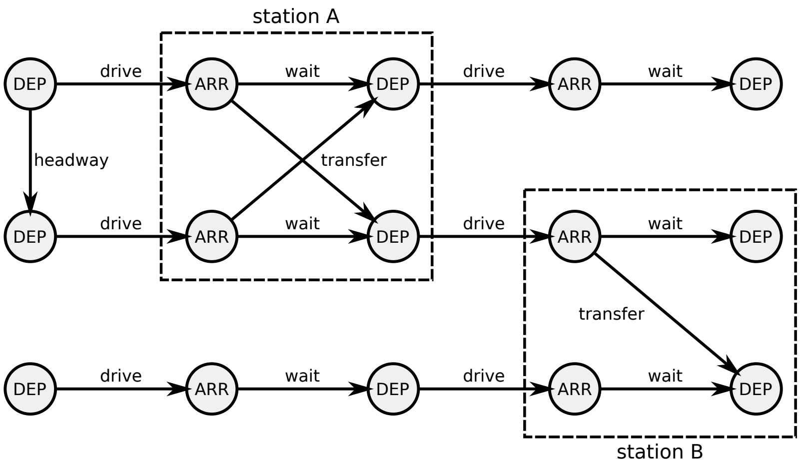


Figure 1 Example event-activity network.

constraints, see [10] for some further modelling features. We present an example EAN with three trains in Figure 1.

For each activity $a \in \mathcal{A}$, we are given an interval $[\ell_a, u_a]$ reflecting a lower and upper bound on its duration. We call $u_a - \ell_a$ the span of activity a. The aim is to assign a time to each event, such that the durations of activities are within the desired time interval. Let π_i denote the time event $i \in \mathcal{E}$ takes place. Due to the periodicity, this means that it also takes place at the time points $\ldots, \pi_i - 2T, \pi_i - T, \pi_i + T, \pi_i + 2T, \ldots$. To reflect this periodicity, each activity $a = (i, j)$ corresponds to a constraint of the form

$$((\pi_j - \pi_i) \mod T) \in [\ell_a, u_a] \,.$$

The modulo operator is linearised by introducing new integer variables z_a, the so-called modulo parameters.

In the original definition of the PESP, no objective function was used. In this paper we follow the widely used approach of minimising slack times. To this end, we assume a weight w_a for each arc to be given, which reflects the penalty that is to be applied to any time unit of slack. In the case of a transfer arc a, w_a might represent the expected number of passengers that desire to use the activity. The resulting node potential formulation then reads as follows.

$$\min \sum_{a=(i,j)\in\mathcal{A}} w_a(\pi_j - \pi_i + Tz_a - \ell_a) \tag{1}$$

$$\text{s.t. } \ell_a \leq \pi_j - \pi_i + Tz_a \leq u_a \qquad \forall a = (i,j) \in \mathcal{A} \tag{2}$$

$$0 \leq \pi_i \leq T - 1 \qquad \forall i \in \mathcal{E} \tag{3}$$

$$z_a \in \{0, 1, 2\} \qquad \forall a \in \mathcal{A} \tag{4}$$

Note that without loss of generality we may assume here that $\ell_a \in [0, T)$. Yet, we may restrict z_a to the values $\{0, 1\}$ only in the case of a constraint where $u_a \leq T$.

An alternative model for this problem is to use variables x_a to express the duration of every activity. In this case, one needs to fulfil that

$$\sum_{a\in C^+} x_a - \sum_{a\in C^-} x_a \equiv 0 \mod T$$

for every oriented cycle (C^+, C^-). It is sufficient to use an integral cycle basis for this purpose, e.g., the fundamental cycles $\mathcal{C}$ induced by some spanning tree $\mathcal{T}$. The resulting cycle-basis formulation is then given as follows.

$$\min \sum_{a \in \mathcal{A}} w_a (x_a - \ell_a) \tag{5}$$

$$\text{s.t. } \ell_a \leq x_a \leq u_a \qquad \forall (i,j) \in \mathcal{A} \tag{6}$$

$$\sum_{a \in C^+} x_a - \sum_{a \in C^-} x_a = T p_C \qquad \forall C \in \mathcal{C} \tag{7}$$

$$p_C \in \mathbb{Z} \qquad \forall C \in \mathcal{C} \tag{8}$$

This formulation can be further strengthened by constraints on p_C, which are known as the Odijk cuts (see [12]). Finding a feasible solution to the PESP is already NP-hard, and finding an optimal solution is considered notoriously difficult.

3 An Iterative Solution Approach

We now describe a new heuristic to find feasible solutions to the periodic timetabling problem with good objective values. It is based on the idea of combining two subprocedures that describe solutions differently. By running both methods alternately, we can find an improvement with one approach when the other approach has become stuck. The first subprocedure uses a heuristic network aggregation that makes use of a standard MIP solver viable. The second subprocedure is the modulo network simplex approach (ModSim). In the following, we describe both of these steps.

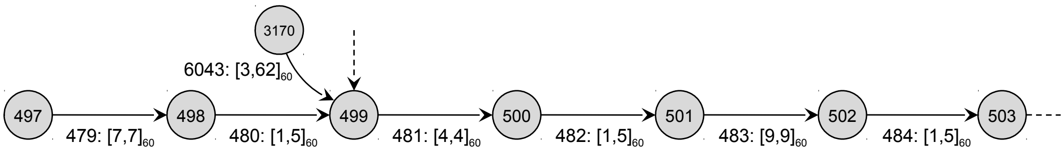
3.1 Aggregation Procedure

We describe preprocessing mechanisms to aggregate and simplify the event-activity network to create a reduced instance, which is smaller and – hopefully – easier to solve. Some of these techniques do not preserve equivalence in the sense that an optimal (partial) solution of the reduced instance can be extended to an optimal (full) solution of the original instance. However, we never affect feasibility, i.e., there is a surjection from all feasible solutions of the original network to the feasible solutions of the reduced network.

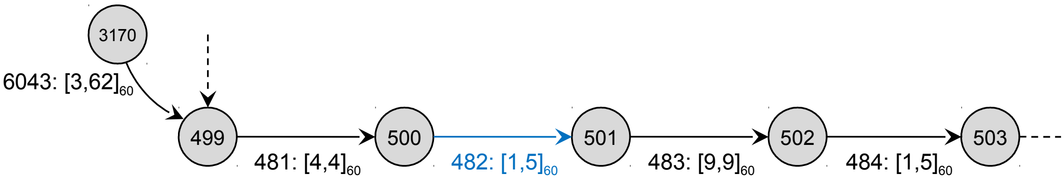
Contraction. There are three ways in which we reduce the initial EAN (see [8]). We illustrate these ideas in Figure 2 where we apply the different variants subsequently to the very same network.

The first two operations are standard graph contractions for which we are able to keep the set of optimal solutions. In the special case of a node i with degree one, we simply remove the only arc a that is incident to i, together with i itself. In the reverse operation, we derive the value for π_i simply such that $x_a = \ell_a$. Observe that doing so, there is a bijection between the optimum solutions of the initial network and of the reduced network.

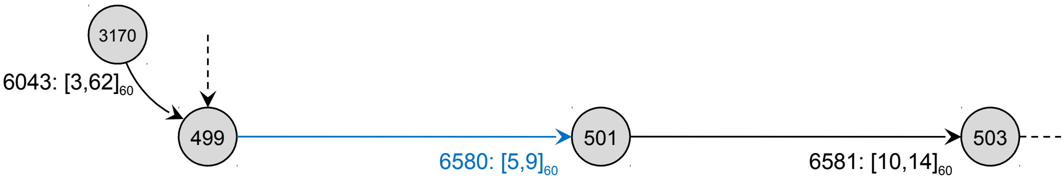
The same holds for the second type of contractions: For a fixed arc $a = (i,j)$, i.e. where $\ell_a = u_a$, we remove a and j. Any arc b that had been incident with j gets "deviated" to i, where we add or subtract ℓ_a to ℓ_b and u_b in the case of b formerly leaving or entering j, respectively. In Figure 2c we contract arc 481 and modify arc 482 by changing its starting node from 500 to 499 and add the value four of the contracted arc to both, its lower and upper bounds. For the reverse operation, we disaggregate node j by simply setting $\pi_j = \pi_i + \ell_a$.



(a) Original network.



(b) After degree-one contraction.



(c) After contraction of fixed arcs.

(d) After (heuristic) degree-two contraction.

Figure 2 Examples for graph contraction in event-activity networks.

The third case in which we apply contractions is a node j with precisely one incoming arc $a = (i, j)$ as well as one outgoing arc b. In this case, we replace j in a with the endpoint of b. Regarding the time constraints, we preserve the set of feasible solutions by adding ℓ_b and u_b to ℓ_a and u_a, respectively, see Figure 2d. Yet, with respect to the objective function, there is a (slight) imprecision, because along the modified arc, for the first units of slack there should apply $\min\{w_a, w_b\}$, whereas $\max\{w_a, w_b\}$ had to apply to the last units of slack. This cannot be expressed in any linear objective function on the modified arc in the reduced network. In our experiments, we heuristically select $\min\{w_a, w_b\}$ as the weight of the modified arc.

Ignoring Light Arcs. Observe that none of the above steps reduces the cyclomatic number $|\mathcal{A}| - |\mathcal{E}| + 1$ of the constraint graph. Yet, it is commonly assumed that this correlates with the computational complexity of PESP instances. Hence we are trying to remove arcs from the constraint graph. Doing so, we must be cautious: If the reduced graph has any feasible solution which can *not* be translated into a solution of the initial network, then the entire consideration of the reduced network would be useless.

Since infeasibility may only arise on an arc a with span $u_a - \ell_a < T - 1$, we focus exclusively on *free* arcs with span $u_a - \ell_a \geq T - 1$. These are arcs which model just waiting times (e.g. of passengers at transfers, of trains during turnarounds, of both during stops) but do not model any strict operational requirement. Thus, such arcs can simply be omitted without affecting the set of feasible solutions.

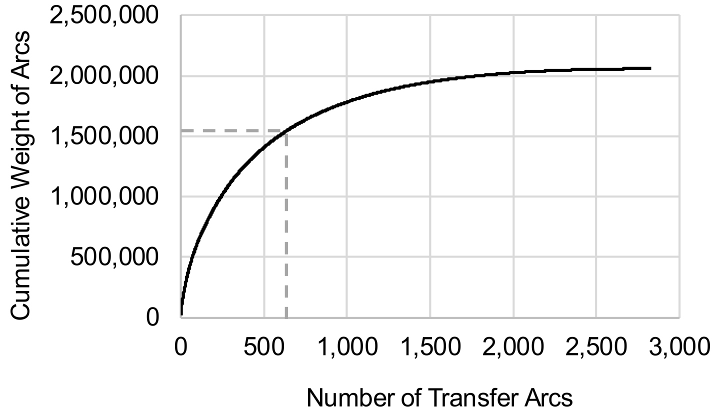


Figure 3 Distribution of the weights w_a of the free arcs in R1L1.

For instances of the PESPlib, the constraint graph even decomposes into cycle-free connected components, which are trivial to solve optimally, when omitting all the free arcs. This is due to the absence of headway or single track constraints in these instances.

Obviously, there is a trade-off: The more free arcs are ignored, the easier becomes the resulting instance to solve. However, solutions to the simplified instance become less significant for the initial problem, because they may induce large waiting times along the ignored arcs.

Hence, we finally investigate how the weights of these free arcs are distributed. The plot in Figure 3 shows that for example the R1L1-instance follows the so-called “Pareto principle”: When removing, e.g., 77.5% of the free arcs (abscissa) we are ignoring just 25% of the initial total weight $W := \sum_{a \in \mathcal{A}: a \text{ is free}} w_a$ of the free arcs (ordinate). Thus, we may significantly simplify an instance while only losing a limited amount of information (i.e., weight).

Our network aggregation procedure is used to generate an instance that is sufficiently small to allow a mixed-integer programming solver to be applied. This is then combined with the ModSim-method as in a ball game, i.e., by giving the solution of one approach as input for the other. The ModSim-method is briefly summarised in the following section.

3.2 Description of the Modulo Network Simplex

We now briefly describe the ModSim-method, and refer to [4] for details. It is based on the observation that there exists an optimal solution to the periodic timetabling problem that is induced by a spanning tree structure $\mathcal{T} = (\mathcal{T}_\ell, \mathcal{T}_u)$ in $\mathcal{N}$. This means that we set $x_a = \ell_a$ for all edges in $\mathcal{T}_l$, and $x_a = u_a$ for all edges in $\mathcal{T}_u$. The duration of all other activities and their modulo parameters are then uniquely determined.

The method performs a local search over the set of spanning tree structures, until it finds a local optimum (i.e., all spanning tree structures that can be reached by exchanging a single arc do not provide a feasible solution with better objective value). This is called the inner loop. It then tries to escape the local optimum using modifications that are not based on a spanning tree structure; e.g., single node cuts or multi node cuts. If an improved solution can be found, a new spanning tree structure is calculated and the method is repeated from the beginning. This is called the outer loop. A schematic description of this method is given in Figure 4.

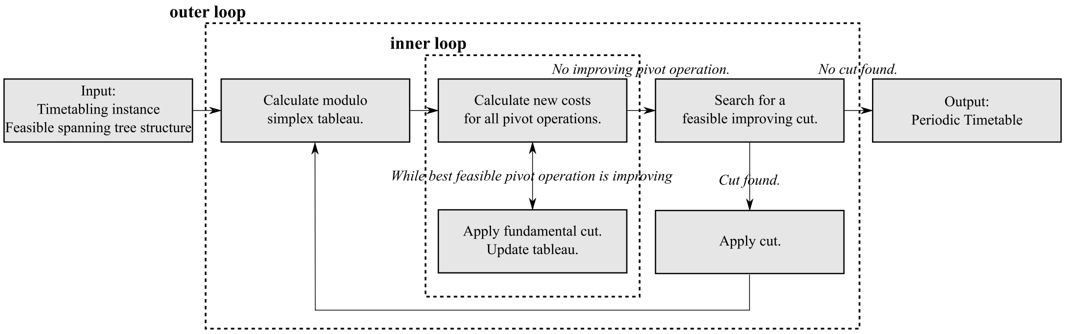


Figure 4 The modulo network simplex procedure (see [4]).

Table 1 Properties of PESPlib instances.

name	nodes	arcs	fixed arcs	other arcs	free arcs
R1L1	3,664	6,385	646	2,912	2,827
R1L2	3,668	6,543	632	2,928	2,983
R1L3	4,184	7,031	758	3,302	2,971
R1L4	4,760	8,528	830	3,788	3,910
R2L1	4,156	7,361	819	3,210	3,332
R2L2	4,204	7,563	822	3,252	3,489
R2L3	5,048	8,286	971	3,918	3,397
R2L4	7,660	13,173	1,501	5,932	5,740
R3L1	4,516	9,145	799	3,576	4,770
R3L2	4,452	9,251	776	3,538	4,937
R3L3	5,724	11,169	1,042	4,496	5,631
R3L4	8,180	15,657	1,480	6,462	7,715
R4L1	4,932	10,262	996	3,764	5,502
R4L2	5,048	10,735	986	3,886	5,863
R4L3	6,368	13,238	1,242	4,898	7,098
R4L4	8,384	17,754	1,573	6,546	9,635

For the purpose of this paper, two characteristics of the ModSim are particularly relevant: Firstly, the use of spanning tree structures means that solutions are encoded in a different way than in the MIP formulation. Secondly, the method can be run for a (practically) arbitrary amount of time, as the search space for the outer loop is too large to be fully explored.

4 Experimental Results

4.1 Setting and Instances

We use the 16 periodic railway timetabling instances from the PESPlib[3], created by the public transport planning software LinTim[4], see also [3]. The size of the event activity networks and the numbers of fixed, free and other arcs is given in Table 1.

To assess the quality of our aggregation method, we performed two different sets of experiments. In the first experiment, we discuss the impact of the aggregation for different

[3] `http://num.math.uni-goettingen.de/~m.goerigk/pesplib/`
[4] `https://lintim.math.uni-goettingen.de/`

Table 2 Graph aggregation statistics for R1L1.

	nodes	arcs	eliminated	heuristic?
original	3,664	6,385	–	–
contract deg one	3,216	5,937	448	no
contract fixed	2,677	5,398	539	no
contract deg two	1,228	3,949	1,449	(yes)
ignore 25%	1,228	1,756	2,193	yes
contract deg one	863	1,391	365	no
contract deg two	501	1,029	362	(yes)

settings on a single instance (R1L1). In the second experiment, we run our iterative method and compare the objective values we find to using only the ModSim approach. Whenever a MIP is solved for PESP, we used a formulation that combines both the node potential variables (π) and the periodic tension variables (x). We consider the modulo parameters (z) on the arcs and exploit the fact that if $a \in \mathcal{T}$, then we may set $z_a = 0$, while losing the initial property that $z_a \in \{0, 1, 2\}$.

4.2 Results of our Preprocessing Method

For our first set of experiments, we used an Intel core i5 2.20GHz with 8GB RAM, and CPLEX 12.7 with memory limit 2GB. We start by summarising the impact of each aggregation step on the instance R1L1 in Table 2. Starting with an initial problem size of 3,664 nodes and 6,385 arcs, we finally come up with a *similar* instance with only 501 nodes and 1,029 arcs.

Recall that the cyclomatic number is only reduced in the step where we – heuristically – ignore the lightest arcs. In this particular setting, we continue until the weights of the ignored arcs sums up to 25% of the initial total free weight W. Notice that by ignoring the lightest arcs, some of the nodes end up with degree one. This is why we apply the corresponding contraction steps anew.

Next we want to identify good ignore ratios by trading simplification (and thus MIP performance) against significance, i.e. reinterpretability. To this end, we simplify the R1L1 PESPlib-instance by ignoring 10% to 70% of the total free weight and apply CPLEX 12.6 for 15 minutes at default parameter settings. By growing the ignore ratio the size of the resulting simplified network decreases – and so does the optimality gap of the CPLEX run on this simplified network, see the column “gap” in Table 3.

But when reinterpreting the solution that CPLEX obtained for the reduced network, back on the initial network, the trade-off becomes obvious: by ignoring more than 20% of the initial total free weight W, the solution for the actual instance R1L1 is getting worse (cf. column “objective”).

Let us annotate that the improvement from the 2013 PESPlib benchmark (37,338,904) down to 36,213,298 (cf. the 30%-row in Table 3) is *not* just due to a version improvement inside the MIP solver: When applying the 2012 version (CPLEX 12.3) with the same parameter setting to the very same reduced MIP file, already this yields a disaggregated solution of value 35,903,663 for the R1L1-instance of the PESPlib. Unfortunately, this computation had only been possible on a machine with an Intel Xeon 3.7GHz and 16GB RAM.

In one further run we spent one hour with CPLEX and set the ignore ratio right between the two best ones of our initial series, thus 25%. The solver is able to find a significantly better solution (33,711,523). In this solution, 29,763,908 units of weighted slack arise along

Table 3 Objective values found for R1L1 by applying different ignore ratios for the free arcs.

ignore	nodes	arcs	Odijk	time	CPLEX gap	objective
10%	772	1,828	yes	900	6.89%	37,918,546
20%	572	1,230	yes	900	5.69%	35,433,189
30%	438	862	yes	900	4.73%	36,213,298
40%	346	610	yes	900	3.36%	36,720,735
50%	257	406	yes	900	1.77%	40,814,013
60%	189	251	yes	900	0.75%	41,843,259
70%	129	136	yes	900	0.00%	46,010,226
25%	501	1,029	no	3,600	4.22%	33,711,523

free arcs (e.g. transfers), the rest appears on arcs with smaller span (e.g. dwell arcs). All the free arcs together show a weighted average slack of 25% of the period time. This is composed of a weighted average slack time of 46% of the period time when restricted to the $2,193$ ignored light arcs – but only 17% of the period time when restricted to the 634 heavier[5] arcs that our heuristic kept for the simplified core problem. Yet, with our combined iterated method we are able to report even better solutions in the next subsection.

4.3 Results of our Iterative Solution Method

For our second set of experiments, we used a 16-core Intel Xeon E5-2670 processor, running at 2.60 GHz with 20MB cache. Processes were pinned to one core. We used CPLEX v.12.6 to solve MIPs, choosing the *MIPemphasis* parameter so that the solver focus is on improving the primal bound.

The aim of this experiment is to compare the ModSim-method as a stand-alone approach with our iterative method. To this end, we allow both methods 8 hours of computation time for each instance, using the same starting solution found through a constraint propagation procedure. In our iterative method, we begin with the network aggregation step and allow CPLEX up to 15 minutes of computation time. We then use the ModSim for 45 minutes and repeat. For the first network aggregation, we ignore 50% of total free weight W. This number is multiplied by 0.6 in each iteration (i.e., in the second iteration, 30% is ignored, then 18%, and so on). This means that the models CPLEX has to solve become larger and harder to solve, but also more detailed and closer to the actual problem.

We present the final objective value of the ModSim approach and the best objective value of our iterative method in Table 4. Our approach outperforms the pure ModSim on each of the 16 instances, by an average of 15.7% (min: 5.3%, max: 22.8%). We improve all current best solutions from PESPlib by at least 10%, in particular on R1L1, which has been approached with other methods.

We give a more detailed view on the progress of the solution methods in Figure 5. Here we compare the current objective value over the 8 hours time horizon between the ModSim-method and our approach. Detailed results for the other instances can be found in the appendix.

In particular for the smaller instances, using CPLEX on the aggregated network can lead to solutions which have a higher objective value than before. As we reduce the number of ignored activities in every iteration, these errors (visible as bumps in the objective value curve) become smaller over time.

[5] Recall that since we apply contractions again after we ignored $2,193$ light free arcs, in the end there are only 623 free arcs remaining in the simplified problem.

Table 4 Objective values

Instance	PESPlib obj.	start obj.	ModSim obj.	iterative obj.	impr. to ModSim	impr. to PESPlib
R1L1	37,338,904	44,486,347	38,523,096	31,194,961	19.0%	16.5%
R1L2	38,248,408	49,197,598	40,616,172	31,682,263	22.0%	17.2%
R1L3	38,612,098	51,614,049	39,308,815	30,535,261	22.3%	20.9%
R1L4	35,955,823	47,185,054	34,350,087	27,893,098	18.8%	22.4%
R2L1	53,708,802	66,714,782	48,822,627	42,502,069	12.9%	20.9%
R2L2	47,836,571	65,925,364	47,003,096	43,068,782	8.4%	10.0%
R2L3	46,530,294	62,152,599	42,179,765	39,942,656	5.3%	14.2%
R2L4	42,848,107	60,588,717	42,811,532	33,063,475	22.8%	22.8%
R3L1	53,299,647	72,124,479	52,936,675	45,483,668	14.1%	14.7%
R3L2	53,441,333	72,434,508	53,680,036	46,228,200	13.9%	13.5%
R3L3	48,707,212	68,215,489	47,014,354	43,039,089	8.5%	11.6%
R3L4	40,597,536	59,064,577	46,044,547	35,547,064	22.8%	12.4%
R4L1	59,225,243	81,570,976	56,935,920	51,650,471	9.3%	12.8%
R4L2	59,292,152	82,131,250	57,892,642	51,965,758	10.2%	12.4%
R4L3	54,975,374	78,797,377	58,075,575	45,881,499	21.0%	16.5%
R4L4	47,140,800	63,464,363	51,128,274	41,163,954	19.5%	12.7%

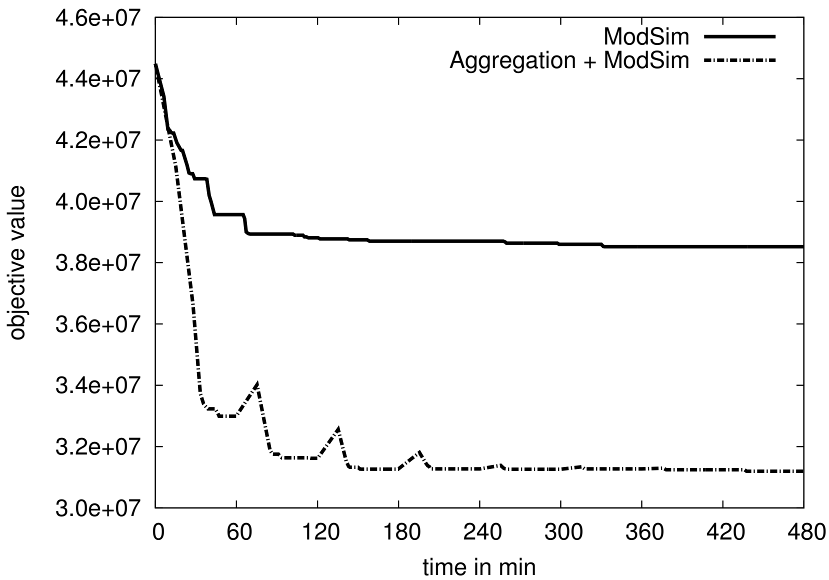


Figure 5 Objective value over time for R1L1.

5 Conclusions

Finding good periodic timetables that offer short travel times for passengers and respect security constraints is a highly relevant public transport planning problem worldwide, but existing solution methods still show an unsatisfactory performance on real-world sized instances. In this paper we presented a new approach to this problem that combines one of the most successful current heuristics, the modulo network simplex method, with a network aggregation step that allows to use a mixed-integer programming solver (CPLEX in our case). As the modulo network simplex method and CPLEX describe solutions differently, it is possible to escape from a local optimum by switching methods. This leads to a significantly improved overall performance. Our approach found solutions that perform on average over 15% better than using the modulo network simplex alone, and improve all current best results that can be found in the PESPlib dataset.

In further research more possibilities to combine the network aggregation with the modulo network simplex heuristic will be explored, including ways to avoid a repetition of solutions between the two methods. Additionally, lower bounds for periodic timetabling problems tend to be weak when a mixed-integer programming solver is used, which leads to a large optimality gap. We will investigate to what extent the network aggregation procedure can be used to produce stronger lower bounds.

Acknowledgment. The authors thank Michel Le (IBM) and Ralf Borndörfer for recently providing us with the CPLEX version of the year 2012 (12.3).

References

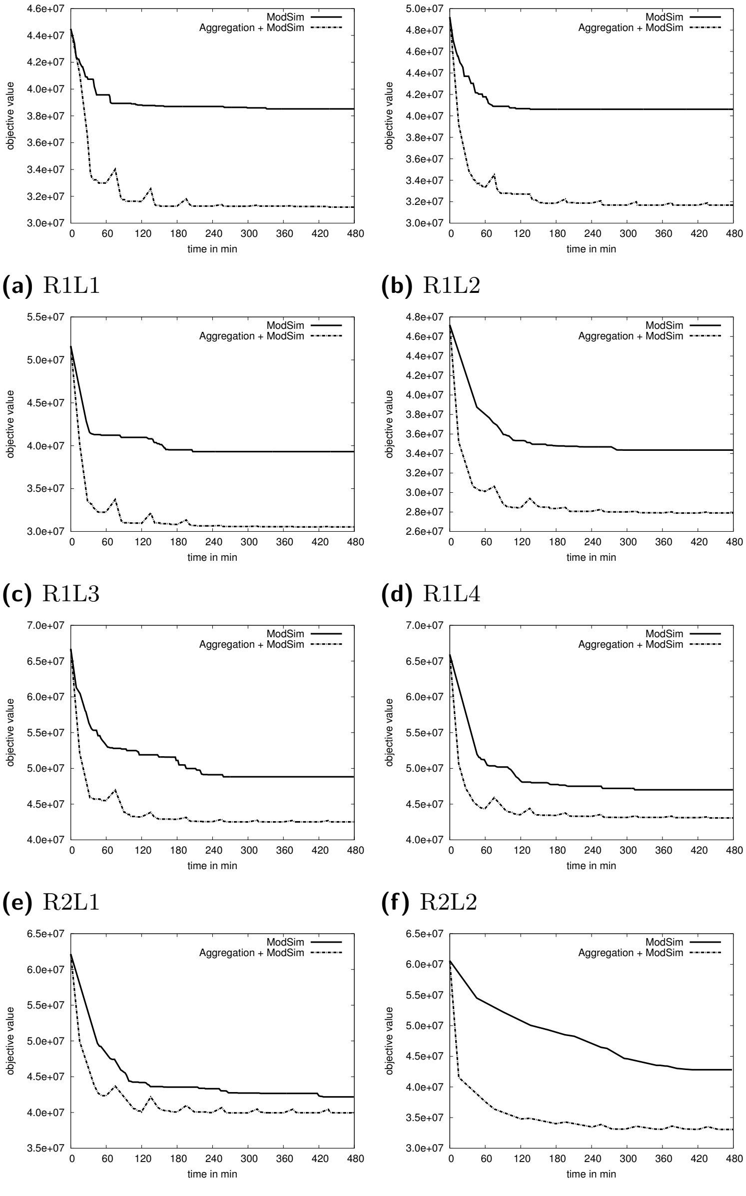
1. Valentina Cacchiani and Paolo Toth. Nominal and robust train timetabling problems. *European Journal of Operational Research*, 219(3):727–737, 2012.
2. Gabrio Caimi, Leo Kroon, and Christian Liebchen. Models for railway timetable optimization: Applicability and applications in practice. *Journal of Rail Transport Planning & Management*, 6:285–312, 2017.
3. Marc Goerigk, Michael Schachtebeck, and Anita Schöbel. Evaluating line concepts using travel times and robustness. *Public Transport*, 5(3):267–284, 2013.
4. Marc Goerigk and Anita Schöbel. Improving the modulo simplex algorithm for large-scale periodic timetabling. *Computers & Operations Research*, 40(5):1363–1370, 2013.
5. Marc Goerigk and Stephan Westphal. A combined local search and integer programming approach to the traveling tournament problem. *Annals of Operations Research*, 239(1):343–354, 2016.
6. Leo Kroon, Dennis Huisman, Erwin Abbink, Pieter-Jan Fioole, Matteo Fischetti, Gábor Maróti, Alexander Schrijver, Adri Steenbeek, and Roelof Ybema. The new Dutch timetable: The OR revolution. *Interfaces*, 39(1):6–17, 2009.
7. Christian Liebchen. Optimierungsverfahren zur Erstellung von Taktfahrplänen. Master's thesis, Technical University Berlin, Germany, 1998. In German.
8. Christian Liebchen. *Periodic Timetable Optimization in Public Transport*. Dissertation.de – Verlag im Internet, 2006.
9. Christian Liebchen. The first optimized railway timetable in practice. *Transportation Science*, 42(4):420–435, 2008.
10. Christian Liebchen and Rolf H Möhring. The modeling power of the periodic event scheduling problem: railway timetables—and beyond. In *Algorithmic methods for railway optimization*, pages 3–40. Springer, 2007.
11. Karl Nachtigall and Jens Opitz. Solving Periodic Timetable Optimisation Problems by Modulo Simplex Calculations. In *8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'08)*, volume 9 of *OpenAccess Series in Informatics (OASIcs)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2008. doi:10.4230/OASIcs.ATMOS.2008.1588.
12. Michiel A Odijk. A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research Part B: Methodological*, 30(6):455–464, 1996.

13 Julius Pätzold and Anita Schöbel. A Matching Approach for Periodic Timetabling. In *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016)*, volume 54 of *OpenAccess Series in Informatics (OASIcs)*, pages 1–15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. `doi:10.4230/OASIcs.ATMOS.2016.1`.

14 Paolo Serafini and Walter Ukovich. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics*, 2(4):550–581, 1989.

A Objective value comparison for each instance

Detailed results for each instance comparing the objective value over time when using ModSim only, and our approach can be found in Figures 6 and 7.



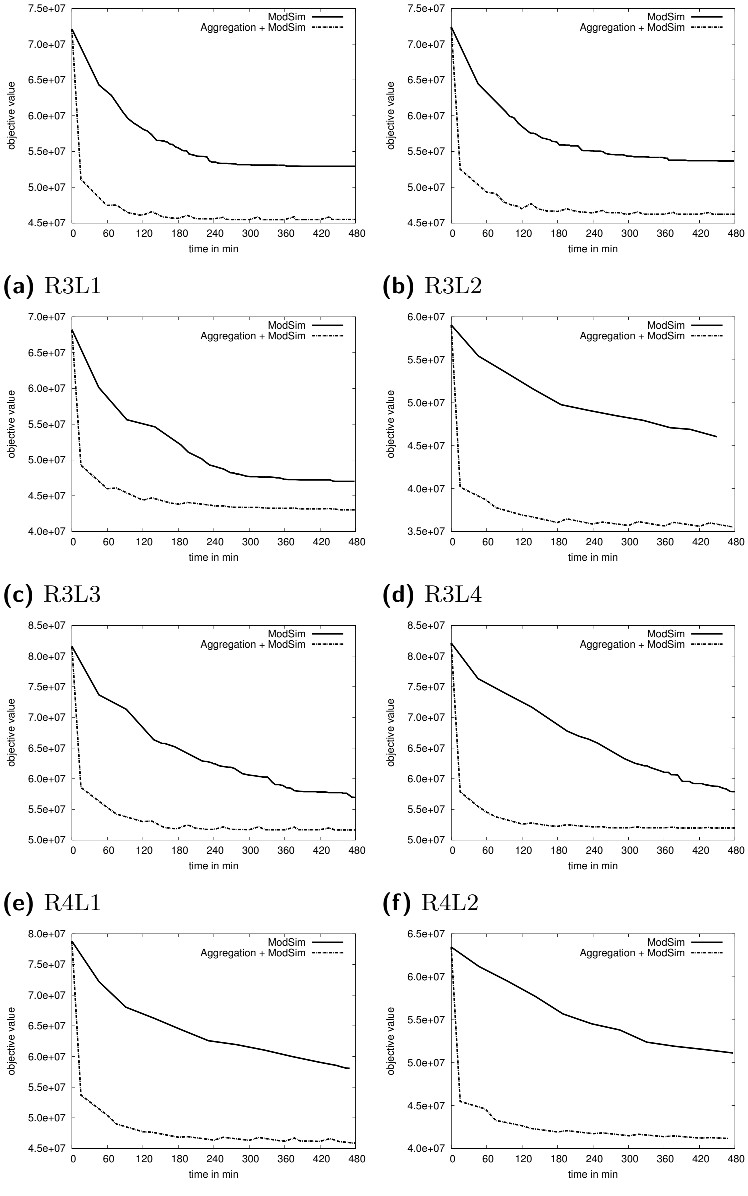
(a) R1L1 **(b)** R1L2

(c) R1L3 **(d)** R1L4

(e) R2L1 **(f)** R2L2

(g) R2L3 **(h)** R2L4

Figure 6 Objective value over time for ModSim and our iterative method.



(a) R3L1 **(b)** R3L2 **(c)** R3L3 **(d)** R3L4 **(e)** R4L1 **(f)** R4L2 **(g)** R4L3 **(h)** R4L4

Figure 7 Objective value over time for ModSim and our iterative method.